The University of Hong Kong

COMP4801 Final Year Project

Final Report

Lie Detector with Machine Learning

Written By: Li Kwong Tsun Ethan
Group Members: Li Kwong Tsun Ethan 3035689116
Chan Lok Him Leo 3035663356
Date of Submission: 18/04/2023
Supervisor: Dr. Chan Kwok Ping
Abstract

The ability to detect lies is vital nowadays because humans tend to deceive others to gain the upper hand in society. Professionals can detect lies, but human errors are inevitably unavoidable. Therefore, the study of lie detection using machine learning has become more prominent. This final year project aims to create a machine learning model for lie detection using deep learning and ensemble learning. At the same time, extend the work of previous research on this topic. More work can be conducted in terms of classifying different modalities and using different machine learning classification techniques, analyzing each combinations’ effectiveness in classifying veracity. This project applies several machine learning classification algorithms, such as Random Forest, Multi-layer Perceptron (MLP), ensemble learning and the VGG-16 model, to train on data extracted from a single dataset. The proposed dataset, Miami University Deception Detection Dataset (MU3D), includes video, audio, and text data, which are extracted and utilized in this project. The three proposed data types are first classified into truth or lie, then the results are combined for the final prediction.

The project concludes with the best and final model ending up at 60.9% accuracy in classifying the veracity of spoken statements. The achieved accuracy is within £2% compared to other models created by other researchers with similar approach. Though, as a binary classifier, there is still much room for improvement. Future work can be done by adding more combinations of modalities and classification algorithms, and removing those that are hindering the final model’s performance. The project team believes that with more experimentation and training data, this project will have a huge impact on improving lie detection algorithms, and possibly break through the state of the art on this matter.
Acknowledgment

I would like to express my gratitude to my supervisor Dr. Chan for fully supporting the team and answering all our questions. He is very kind and always gives us plenty of timely feedback and guidance on our work. I want to thank my groupmate Leo for being a hardworking teammate and always suggesting new ideas to improve our work. I would also like to thank Ms. Emily Paige Lloyd, the creator of the Miami University Deception Detection Database (MU3D), for permitting us to use her dataset for our project. Finally, I would like to thank my family for always supporting me during the long nights working on this project.
Table of Contents

Abstract ........................................................................................................................................... 2
Acknowledgment ............................................................................................................................ 3
List of Figures ................................................................................................................................. 6
List of Tables .................................................................................................................................. 8
Abbreviations ................................................................................................................................ 10

Chapter 1 – Introduction ........................................................................................................... 11
  1.1 Background ......................................................................................................................... 11
  1.2 Motivation ........................................................................................................................... 12
  1.3 Objectives and Contribution............................................................................................ 12
  1.4 Outline of the Report ........................................................................................................... 12

Chapter 2 – Literature Review ............................................................................................... 13
  2.1 Human Lie Detectors .......................................................................................................... 13
  2.2 Polygraph Lie Detectors ...................................................................................................... 13
  2.3 Machine Learning Lie Detectors ......................................................................................... 14
  2.3 Multi-modal Machine Learning Lie Detectors.................................................................... 15
  2.4 Research Gap....................................................................................................................... 16

Chapter 3 – Methodology ........................................................................................................... 17
  3.1 Overview of Development Procedures ............................................................................... 17
  3.2 Developmental Technologies .............................................................................................. 18
  3.3 Model Training Procedures ................................................................................................. 19
    3.3.1 Phase 1 - Data Collection (MU3D) .............................................................................. 19
    3.3.2 Phase 2 – Data Extraction and Pre-processing ............................................................. 20
    3.3.3 Phase 3 – Classification ................................................................................................ 28
    3.3.4 Phase 4 – Ensemble ...................................................................................................... 29

Chapter 4 – Experiments and Results ....................................................................................... 30
  4.1 Experiments with Original Frame Module ......................................................................... 30
    4.1.1 Experiment 1 – 20 Frames with VGG16 ...................................................................... 30
    4.1.2 Experiment 2 – 20 Frames with different ML Classifiers ............................................ 31
    4.1.3 Experiment 3 – Optimizing MLP Layers ..................................................................... 33
    4.1.4 Experiment 4 – Data Augmentation ............................................................................. 35
    4.1.5 Experiment 5 – New Frame Extraction Method & PCA............................................. 36
List of Figures

Figure 3.1 – Overview of Model Structure ...............................................................16
Figure 3.2 – Example of MU3D .............................................................................18
Figure 3.3 – Illustrated process of frame extraction method 1 ...............................19
Figure 3.4 – Illustrated process of frame extraction method 2 ...............................20
Figure 3.5 – The VGG-16 architecture .................................................................21
Figure 3.6 – Example of an image cropped to facial image using autocrop ...............22
Figure 3.7 – An image that cannot be cropped by autocrop .................................23
Figure 3.8 – Illustrated process of face extraction ...............................................24
Figure 3.9 – An expression classification example for a facial image extracted from MU3D ................................. 25
Figure 3.10 – Visualized conversion from audio to MFCC ...................................26
Figure 4.1 – Results of Original Frame Module Experiment 1 ..............................29
Figure 4.2 – Results of Original Frame Module Experiment 2 with MLP ...............30
Figure 4.3 – Results of Original Frame Module Experiment 2 with SVM ..........31
Figure 4.4 – Results of Random Forest Classifier in Original Frame Module Experiment 6 ..........................................................36
Figure 4.5 – Results of KNN Classifier in Original Frame Module Experiment 6 ..........37
Figure 4.6 – Results of MLP Classifier in Original Frame Module Experiment 6 ..........38
Figure 4.7 – Results of KNN Classifier in Facial Image Module Experiment 3 ..........41
Figure 4.8 – Results of Random Forest Classifier in Facial Image Module Experiment 3 ..........42
Figure 4.9 – Results of MLP Classifier in Facial Image Module Experiment 3 ..........43
Figure 4.10 – Results of KNN Classifier in Audio Module Experiment 1 ...............44
Figure 4.11 – Results of Random Forest Classifier in Audio Module Experiment 2 ..........45
Figure 4.12 – Top 20 most frequent words in ngram_range (1,1) .................46
Figure 4.13 – Top 20 most frequent words pairs in ngram_range (2,2) ..........................47

Figure 4.14 – Pipeline for Text Module Experiment 3 ..................................................48

Figure 4.15 – The results of Ensemble Experiment 2 .................................................51

Figure 4.16 – Screenshot of the GUI Input fields .........................................................56

Figure 4.17 – Example Output of Final Product .........................................................58
List of Tables
Table 2.1 – Overview of related works .................................................................13
Table 2.2 – Overview of multi-modal machine learning lie detectors ......................14
Table 2.3 – Accuracies of different combinations of classifiers in Bag-of-Lies ..............14
Table 3.1 – List of combinations of modalities and machine learning algorithms ........27
Table 3.2 – Example of ensemble method 1 ..........................................................28
Table 3.3 – Example of ensemble method 2 ..........................................................28
Table 3.4 – Example of ensemble method 3 ..........................................................28
Table 4.1 – Accuracy results of Original Frame Module Experiment 3 ....................32
Table 4.2 – Accuracy results of Original Frame Module Experiment 4 ....................34
Table 4.3 – The accumulated explained variance for each PCA value (Original Frame) ....35
Table 4.4 – Accuracy results of Facial Image Experiment 1 ....................................39
Table 4.5 – The accumulated explained variance for each PCA value (Facial Image) ......40
Table 4.6 – The results for different ngram_range values in Text Module Experiment 1 ....47
Table 4.7 – The results for different use_idf values in Text Module Experiment 2 ........48
Table 4.8 – Results of combinations of modalities and machine learning algorithms ....49
Table 4.9 – Failed example of ensemble method 1 ................................................50
Table 4.10 – The trained weights for each module model ......................................52
Table 4.11 – Relationship between Classifiers 1 ....................................................54
Table 4.12 – Relationship between Classifiers 2 ....................................................54
Table 4.13 – Accuracy of different combinations of modalities ................................55
Table A – List of python libraries used during development ......................................64
## Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MU3D</td>
<td>Miami University Deception Detection Database</td>
</tr>
<tr>
<td>PT</td>
<td>Positive Truth</td>
</tr>
<tr>
<td>PL</td>
<td>Positive Lie</td>
</tr>
<tr>
<td>NT</td>
<td>Negative Truth</td>
</tr>
<tr>
<td>NL</td>
<td>Negative Lie</td>
</tr>
<tr>
<td>VGG-16</td>
<td>Machine Learning Model created by Visual Geometry Group from Oxford</td>
</tr>
<tr>
<td>ML</td>
<td>Machine Learning</td>
</tr>
<tr>
<td>MLP</td>
<td>Multi-layer Perceptron</td>
</tr>
<tr>
<td>SVM</td>
<td>Support Vector Machine</td>
</tr>
<tr>
<td>KNN</td>
<td>K-nearest Neighbors algorithm</td>
</tr>
<tr>
<td>EEG</td>
<td>Electroencephalogram</td>
</tr>
</tbody>
</table>
Chapter 1 – Introduction

This project aims to build a machine learning model for lie detection in videos. In the process, learning and analyzing the effectiveness of each combination of modality and classification algorithms on this matter. This chapter describes the basic information of the project, including the background, motivation, objectives, deliverables, and outline of this report.

1.1 Background

Lying is the act of presenting an intentionally false statement, mainly to gain an advantage over something or someone. Whether it is an un­harmful white lie to a young one about the existence of Santa Claus, or a malicious lie about fake previous experiences during a job interview, everyone has lied in their life. It is an accepted fact that humans lie and have been lying ever since language was invented. In fact, surveys have shown that an average person lies 4 times every day [1].

With everyone consciously living in lies, developing a sense of skepticism becomes natural in some ways. Before the advancements in technology, people have noticed common behaviors when someone is lying. For example, having a fake smile, speaking in a higher tone, or blinking faster [2]. These are the natural responses to nervousness, which is usual when one is lying. People who can catch on to these little behaviors can sometimes determine if someone is lying, however, it is very unreliable as nervous behaviors are not the same for everyone.

To objectively determine if someone is lying, it is impossible to only rely on human instincts. Therefore, a polygraph machine is used for lie detection. These machines can detect the suspects’ blood pressure, pulse, and skin conductivity. Professionals can tell whether the suspect is lying by analyzing the fluctuations in the data. However, there are plenty of disadvantages to using polygraph machines. Details will be further discussed in the next chapter.

To avoid some of the problems related to polygraph lie detector machines, some have created machine learning models to analyze whether someone is telling the truth or not. By using machine learning, modality will not be restricted to regular blood pressure, pulse, or skin conductivity analysis. Some researchers used common data types like video, audio, or text for their models, while others used more advanced data types like electroencephalogram and eye gaze technology [3,4,5].
1.2 Motivation
Although plenty of research has been done in this field, there is still more to be discovered and documented. Previous researchers used multiple data types to train their lie detector model. With multiple modalities, each model is trained separately and combined at the end for analysis and creation of the final model. However, more research can be done by using different machine learning classification algorithms. Additionally, more studies of the final results should be conducted, such as which algorithm or modality works best for lie detection. Furthermore, the project team aims to use common and accessible data types like videos instead of advanced data types so that any ordinary person with a camera can utilize the final model.

1.3 Objectives and Contribution
This project aims to create a binary classification model for lie detection in videos, while serving as an extension to previous research on this topic. The project team proposes to extract multiple accessible features from the videos, such as frames, audio, and transcriptions. Each modal is trained and classified separately, and the results are later combined using ensemble learning. Different machine learning algorithms will be tested and analyzed while training with each different data type. The documented results and the final model with the best accuracy will serve as the deliverables for this project. This project can deepen the world’s understanding of lie detection using machine learning, which is a solid stepping-stone for creating an accurate real-time lie detector in future studies.

1.4 Outline of the Report
Chapter 1 serves as the introduction to this project. It explains the background and motivation of the project, as well as the team’s aims and goals. Chapter 2 reviews past research conducted on this topic, discussing the advantages and weaknesses of their methodologies and results. Chapter 3 will discuss the methodology of this project. This chapter proposes a training dataset and illustrates how features are extracted and classified. Additionally, it will explain all the technology used in the project’s development. Chapter 4 describes the final product and the experiments conducted to achieve this result. Chapter 5 concludes the project’s findings, discusses the difficulties and limitations, as well as suggesting future work and improvements.
Chapter 2 – Literature Review

This chapter discusses the evolution of lie detection over technological advancements. Some previous work on lie detection with machine learning will be discussed, analyzing their significance and shortcomings, as well as how they influence this project.

2.1 Human Lie Detectors
As mentioned in the previous chapter, humans can sometimes detect deception by basic instincts. However, Bond and DePaulo [6] concluded the results from 206 documents and 24483 judges and found that an ordinary person achieves an average of 54% accuracy in determining deception, with 47% lies correctly classified as deceptive and 61% truths correctly classified as non-deceptive. For professional lie catchers like law enforcement personnel, only a small increase to an average accuracy rate of 55.91% is recorded by Vrij [7].

2.2 Polygraph Lie Detectors
Bond and DePaulo’s research show that relying on human instincts to discriminate truth from lies is insufficient, therefore lie detecting machines are created. The most common and accurate lie detectors used by the authorities are polygraph machines, invented in 1921. These machines can detect the suspects’ blood pressure, pulse, and skin conductivity. Professionals can tell whether the suspect is lying by analyzing the fluctuations in the data. [8]

According to Gaggioli’s study [9], polygraph machines used by a professional can achieve 81% to 91% accuracy in detecting. However, problems may arise with feasibility. Firstly, only trained professionals can utilize polygraph machines. An ordinary person cannot tell accurately whether the suspect is lying by looking at the graphs without proper training. Secondly, the suspect must physically consent to the process and be analyzed. Thirdly, access to polygraph machines is difficult. These machines are not commercial products and are mainly available to the authorities only. Fourthly, it is time-consuming and labor-intensive to conduct lie detection tests if there are many suspects to be questioned. Fifthly, trained suspects who understand how these machines function can easily trick them. Sixthly, opposite to point number five, some subjects may be nervous during the test, which may cause inaccurate false positive results. Therefore, these machines are deprecated in the real world, and are only usually found in fiction or movies.
2.3 Machine Learning Lie Detectors

Machine learning for lie detection avoids most issues with polygraph machines. For example, no trained professionals are required to analyze the results in real-time. Once the machine learning models are trained, they are easy to use and readily available. Moreover, machines can do bulk predictions, saving time and effort. Most beneficially, other possibly interesting and informative modalities can be used for analysis instead of just blood pressure, pulse, and skin conductivity.

Table 2.1 shows an overview of some related works in lie detection with machine learning.

<table>
<thead>
<tr>
<th>Title</th>
<th>Modality</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single-trial lie detection using a combined fNIRS-polygraph system</td>
<td>Brain’s hemoglobin</td>
<td>86.5%</td>
</tr>
<tr>
<td>Single-trial lie detection using a combined fNIRS-polygraph system</td>
<td>signals</td>
<td></td>
</tr>
<tr>
<td>Constructing the lie detection system with fuzzy reasoning approach</td>
<td>Electroencephalogram</td>
<td>89.5%</td>
</tr>
<tr>
<td>Constructing the lie detection system with fuzzy reasoning approach</td>
<td>(EEG)</td>
<td></td>
</tr>
<tr>
<td>Machine Learning-Based Lie Detector Applied to Novel Annotated Game</td>
<td>Facial Image</td>
<td>57.44%</td>
</tr>
<tr>
<td>Dataset [12]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ReLiDSS: Novel lie detection system from speech signal [13]</td>
<td>Speech Audio</td>
<td>88.23%</td>
</tr>
<tr>
<td>Covert lie detection using keyboard dynamics [14]</td>
<td>Keyboard dynamics</td>
<td>90%</td>
</tr>
<tr>
<td>Using machine learning for lie detection: classification of human</td>
<td>Facial Muscle Movement</td>
<td>76.2%</td>
</tr>
<tr>
<td>visual morphology [15]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2.1: Overview of related works

With the help of machine learning, it is possible to utilize modalities that are difficult for humans to manually analyze for classification. Rodriguez-Diaz’s and H. Nasri’s [12,13] research used easily accessible data such as facial image and speech audio. On the other hand, Monaro’s and Samin’s [14,15] work used niche data like keyboard dynamics and facial muscle movements, while Bhutta’s and Lai’s [10,11] models used advanced brain signals as data. As shown from the accuracies in Table A, the results were satisfactory, with most models at >80% accuracy.

However, the downside to these models is that they are of single modality. The accuracy of the model is highly dependent on the input data. For example, a speech audio model would be
inaccurate if bad quality audio is given as input. A solution to this would be to use a multi-modal approach.

2.3 Multi-modal Machine Learning Lie Detectors

The advantage to multi-modal machine learning is that if one modal performs worse than expected, other modalities can pull the weight and attempt to produce the correct prediction.

Table 2.2 shows a list of multi-modal machine learning on lie detectors.

<table>
<thead>
<tr>
<th>Title</th>
<th>Modalities</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deception detection using a multimodal approach</td>
<td>Language use, physiological response, thermal sensing</td>
<td>/</td>
</tr>
<tr>
<td>[16]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Verbal and Nonverbal Clues for Real-life Deception Detection [17]</td>
<td>Verbal and Nonverbal Clues</td>
<td>77-82%</td>
</tr>
<tr>
<td>Deception Detection using Real-life Trial Data [4]</td>
<td>Transcription and gestures</td>
<td>60-75%</td>
</tr>
</tbody>
</table>

Table 2.2: Overview of multi-modal machine learning lie detectors

The models above use multiple modalities and combine the results using ensemble learning. The accuracies of these multi-modal models are worse than the models listed in Table 2.1. This is mainly because most multi-modal models use video data as input, whose accuracy is generally lower compared to advanced datatypes like brain signals, and not because of the models being multi-modal.

Table 2.3 shows the effect of combining individual modalities for the Bag-of-Lies model [3].

<table>
<thead>
<tr>
<th>Modality</th>
<th>Best Average Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Only Video</td>
<td>56.20%</td>
</tr>
<tr>
<td>Only Audio</td>
<td>53.24%</td>
</tr>
<tr>
<td>Combination</td>
<td>Accuracy</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>Only EEG</td>
<td>58.71%</td>
</tr>
<tr>
<td>Only Gaze</td>
<td>61.70%</td>
</tr>
<tr>
<td>EEG + Gaze</td>
<td>62.22%</td>
</tr>
<tr>
<td>EEG + Audio</td>
<td>61.69%</td>
</tr>
<tr>
<td>EEG + Video</td>
<td>60.20%</td>
</tr>
<tr>
<td>Gaze + Audio</td>
<td>63.69%</td>
</tr>
<tr>
<td>Gaze + Video</td>
<td>62.19%</td>
</tr>
<tr>
<td>Audio + Video</td>
<td>60.68%</td>
</tr>
<tr>
<td>Gaze + Video + EEG</td>
<td>62.70%</td>
</tr>
<tr>
<td>Gaze + Audio + EEG</td>
<td>63.21%</td>
</tr>
<tr>
<td>Audio + Video + EEG</td>
<td>63.18%</td>
</tr>
<tr>
<td>Gaze + Video + Audio</td>
<td>64.69%</td>
</tr>
<tr>
<td>All Four</td>
<td>66.17%</td>
</tr>
</tbody>
</table>

Table 2.3: Accuracies of different combinations of classifiers in Bag-of-Lies

As seen from Table 2.3, by combining different modalities, the accuracy of the model increases. The accuracy using all four modalities comes up to 66.17%, which is a 10-20% increase compared to only one modality. This is an expected result achieved by ensemble learning.

2.4 Research Gap

There are plenty of research in lie detection using machine learning, though there are still room for improvement. This project can fill this research gap.

1. This project aims to make a lie detection model using easily accessible data as input such as videos. Compared to other research, most of them use hard to obtain data like brain signals or niche data, therefore collecting new data for testing becomes difficult.
2. With video data as input, it is possible to create a real-time lie detection model. This is because recording a video then importing it to the model is not complicated.
3. This project’s model is multi-modal and utilizes ensemble learning. Each modal is separated into their own module. Future work can be done by simply adding a new module for a new modal if needed, which makes the product easily extendable.
Chapter 3 – Methodology

This chapter presents the methodology of creating the lie detector with machine learning. The overview of the development procedures will first be covered. Then, presents the main technologies and techniques chosen for this project. Next, an overview of the model training procedure will be further explained.

3.1 Overview of Development Procedures

There are 4 phases in the development procedures. Phase 1 is the data collection phase. The project team has outsourced training and testing data for the model. Phase 2 is the data pre-processing phase. The datasets require pre-processing such as feature extraction. Phase 3 is the classification phase. The team will use several different classification algorithms on different data modalities. Phase 4 is the ensemble phase. All the best classifiers for each modality will be fused together, creating the final model.

Figure 3.1 - Overview of Model Structure

Figure 1 describes the overview of the finalized model structure. The project team will extract four features from the proposed database: original video, facial image, audio, and transcript. Then, each feature is trained separately and fused at the end to create the final model. Details are explained in the following sections (section 3.3).
3.2 Developmental Technologies

This section discusses the project’s main developmental technologies, including the proposed programming language and developmental platforms.

**Python**

Python is a programming language with plenty of community-made libraries for developing machine learning applications [18]. The project team uses Python instead of other programming languages because it provides many useful machine learning libraries like TensorFlow, Keras, and Scikit. It is also handy for data manipulation with libraries like numpy and pandas.

**Google Drive**

Google Drive is a file storage and synchronization service developed by Google [19]. It allows users to store files in the Google Cloud Platform and share files with other users. The team decided to store all data used for the project into Google Drive because of its convenience in sharing and accessing data.

**Google Colab**

The Google Colaboratory is a data analysis and machine learning tool created by Google for data scientists or AI researchers to have a shared space to execute intensive Python code [20]. It connects to the Google Cloud Platform and allows users to share work and collaborate with others. The team chose this platform for the project’s development because of its convenience in sharing python code and connecting to Google Drive where the datasets are stored.

**AssemblyAI**

AssemblyAI is built on the latest state-of-the-art AI research to offer production-ready, scalable, and secure AI models through a simple API [21]. It provides an accurate speech recognition model for our project. This is useful if the provided input data does not consist of a transcription, then AssemblyAI’s speech recognition model can quickly generate a transcription for the video, which can be used in the text classification.
3.3 Model Training Procedures
This section describes the details of the entire model training process. The four main phases are:

1. Phase 1 – Data Collection (MU3D)
2. Phase 2 – Data Extraction and Pre-processing
3. Phase 3 – Classification
4. Phase 4 – Ensemble

3.3.1 Phase 1 - Data Collection (MU3D)
For the training dataset, the project will use the Miami University Deception Detection Dataset (MU3D) created by Ms. Emily Paige Lloyd and her students [22]. This dataset contains videos of 80 individuals lying and telling the truth. Participants were asked four questions:

1. Positive Truth (PT) – Describe a person you know who you genuinely like, talk about why you like that person, and describe their positive qualities.
2. Negative Lie (NL) – Describe the same person as above, but this time lie and describe that person as if you dislike them and describe their negative qualities.
3. Negative Truth (NT) – Describe a person you know who you genuinely dislike, talk about why you dislike that person, and describe their negative qualities.
4. Positive Lie (PL) – Describe the same person as above, but this time lie and describe that person as if you truly like them and describe their positive qualities.

Figure 3.2 - Example of MU3D
Figure 2 shows an example of a set of data in MU3D. “BF” in the video name represents “Black Female”. The number “001” means that this is the first participant in the black female group. The remaining shows the veracity type of the question’s response (PT/NL/NT/PL). There are 80 participants, with four videos each in the same format as illustrated above.
The main reason the MU3D was selected for the project is that the dataset is very well documented. For each video clip, the complete transcription is recorded. This allows the project team to conveniently use the transcript as a data type for training. The average accuracy across raters who viewed the video is also indicated, which allows the team to compare the accuracy of humans to the final model easily.

3.3.2 Phase 2 – Data Extraction and Pre-processing

Features are extracted from the MU3D dataset into four different datasets. The methods of pre-processing are described below.

3.3.2.1 Frame Extraction

Each MU3D video consists of 700 to 1200 frames, though not every frame will be extracted. Since extracting every frame and storing them would take too much time and storage space, the project team has decided not to do so. Instead, the following 2 methods were tested for frame extracted. The effect of the 2 methods on the model will be further discussed in the next chapter.

Method 1: Split the video into 20 sets equally and extract the first frame from each set. This results in 20 frames extracted for each video. Consider a 1000-frame video, the following figure illustrates the frame extraction process.

Figure 3.3: Illustrated process of frame extraction method 1
Method 2: Split the video into 10 buckets equally. Extract the 1\textsuperscript{st} frame from each bucket to create set 1. Then extract the 2\textsuperscript{nd} frame from each bucket to create set 2. Repeat the process up to the 20\textsuperscript{th} frame for each bucket. This results in 20 sets of 10 frames each, which is 200 total frames extracted per video. Consider a 1000-frame video, the following figure illustrates the frame extraction process.

Figure 3.4: Illustrated process of frame extraction method 2
3.3.2.2 Frame Feature Extraction (VGG-16)

The features in each frame are extracted by the pre-trained VGG-16 model. The VGG-16 model is a Convolutional Neural Network (CNN) model created by the Oxford Visual Geometry Group [23]. The VGGnet architecture has 41 layers, consisting of 16 layers with learnable weights. It is pretrained on more than a million images from the ImageNet database [24] and it is capable of classifying images into 1000 object categories.

Figure 3.5: The VGG-16 architecture [25]

Figure 3.5 shows the full structure of the VGG-16 network. For this project’s purposes, the feature extraction layers and classifier layers will be separated. Therefore, all layers before the fully connected layers are utilized in this project for feature extraction.

The input of the network is a 224x224 image. If the input image is not 224x224, the network will resize the image to the correct dimensions before the convolution process. In this project, the input images are of any dimensions and the output will be of dimensions 7x7x512 (the output of the last max pooling layer). Since the output dimensions are quite large, with 25088 features for each image, PCA is required to reduce the number of features efficiently. The details of the PCA process will be discussed in the next chapter.
3.3.2.3 Facial Image Extraction

The autocrop library is a community-made face image crop algorithm built by leblanefg [26]. This algorithm can detect faces in images and automatically crop them out. Given the proper parameters, the model can use this algorithm for facial image extraction.

The cropper is defined with the following parameters:

- face_percent = 95
- height = 224
- width = 224

![Figure 3.6: Example of an image cropped to facial image using autocrop](image.png)

An automatic face cropper is very useful since there are a lot of image cropping to be performed. However, the downside to using an automatic algorithm is that it does not always detect a face in each image, and therefore is not able to perform the crop. Out of 320 videos in MU3D, 231 videos are cropped with >90% face detection success rate (>90% of images in the video can be cropped into facial images). 79 videos had 20-90% face detection success rate, but still have acceptable number of face crops. The algorithm had a harder time detecting faces in the remaining 10 videos, with less than 200 faces extracted from each video (all with <20% face detection success rate). The low face detection success rate is most likely due to the bad quality of the video, or the subject’s face is out of frame, or the subject’s glasses hindering the detection algorithm, or the outline of the subject’s face is not clear enough.
Figure 3.7: An image that cannot be cropped by autocrop

Figure 3.7 shows an image that the autocrop algorithm cannot identify a face within, even though it is easy for humans to identify. In fact, the whole video that this image was extracted from did not have a single detectable face according to the autocrop algorithm. Since it would be too much effort to manually crop each problematic video, for the project’s purposes, these videos are discarded from the facial classification.

Similar to Method 2 of frame extraction (see section 3.3.2.1), the facial images are grouped into 20 sets of 10 frames each. Since it is impossible to know which and how many images can be cropped in each video, the team opted to first crop every possible image first. The cropped images are 224x224, therefore does not occupy much storage space, which makes this action possible. After each image is cropped and saved, the total number of faces cropped per video is evident. Since at least 200 frames are needed for Method 2 of frame extraction to work, the 10 videos with less than 200 faces extracted are discarded from the next steps. The remaining 310 videos can proceed as follows. Consider a 1000-frame video with an 80% face detection success rate, the following figure illustrates the extraction process.
Figure 3.8: Illustrated process of face extraction
3.3.2.4 Facial Image Pre-processing

The project proposes 2 methods for facial image pre-processing.

Method 1: Utilize the feature extraction model of VGG-16, like the frame feature extraction method. (See section 3.3.2.2)

Method 2: First use a pre-trained expression classification model to classify the expression for each image. Then take the average of the prediction probability of faces in a video. The averaged array will be the new feature array.

A pre-trained Expression Recognition model will be used in method 2. The model created by WuJie1010 is a CNN based torch implementation on facial expression recognition, achieving 73.112% (state-of-the-art) in FER2013 and 94.64% in CK+ dataset [27].

![Classification results](image)

**Figure 3.9:** An expression classification example for a facial image extracted from MU3D

Figure 3.9 shows a classification example for a facial image extracted from MU3D, made by the proposed Expression Recognition model. It can be classified into 7 categories, namely angry, disgust, fear, happy, sad, surprise and neutral. To create a feature array for further classifying into truth or lie, all faces from each video are classified into expressions, and the results are averaged, resulting in an array of 7 features per video.

The effectiveness of method 1 and 2 will be discussed further in the next chapter.
3.3.2.5 Audio Pre-processing

Each video has its own audio track. The audio track is extracted by using a simple python script with the python library moviepy. The Mel Frequency Cepstral Coefficients (MFCC) are generated from the extracted audio. These are a small set of features which concisely describe the overall shape of a spectral envelope [28]. In this project, 50 MFCC features are extracted. The following figure displays an example of an audio clip from a MU3D video extracted into MFCC.

![50 MFCC Features](image)

Figure 3.10: Visualized conversion from audio to MFCC

3.3.2.6 Transcription Pre-processing

The transcription for each video can be read from the given excel documentation in the MU3D dataset. The text data is further processed with cleaning, tokenization, counting and normalization. Text cleaning is done by transforming all characters to lower case and removing any punctuation. This is to prepare for tokenization and counting, which are done by Scikit’s CountVectorizer method. Normalization is done by Scikit’s TfidfTransformer method. Different parameters are explored in both transformers, which will be further discussed in the next chapter.
3.3.3 Phase 3 – Classification

After the 4 datatypes have been pre-processed, the next step is to train the classifier to predict between truth or lies. The team has experimented on different types of machine learning algorithms with different types of data. The following table shows the combination of modalities and algorithms that were experimented with.

<table>
<thead>
<tr>
<th>Modality</th>
<th>Machine Learning Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original Image</td>
<td>Multilayer Perceptron (MLP)</td>
</tr>
<tr>
<td></td>
<td>VGG-16</td>
</tr>
<tr>
<td></td>
<td>Support Vector Machine (SVM)</td>
</tr>
<tr>
<td></td>
<td>Random Forest</td>
</tr>
<tr>
<td></td>
<td>KNN</td>
</tr>
<tr>
<td>Facial Image</td>
<td>Multilayer Perceptron (MLP)</td>
</tr>
<tr>
<td></td>
<td>Logistic Regression</td>
</tr>
<tr>
<td></td>
<td>Support Vector Machine (SVM)</td>
</tr>
<tr>
<td></td>
<td>Decision Tree</td>
</tr>
<tr>
<td></td>
<td>KNN</td>
</tr>
<tr>
<td></td>
<td>AdaBoost</td>
</tr>
<tr>
<td></td>
<td>Random Forest</td>
</tr>
<tr>
<td>Audio</td>
<td>Random Forest</td>
</tr>
<tr>
<td></td>
<td>KNN</td>
</tr>
<tr>
<td>Transcription</td>
<td>Support Vector Machine (SVM)</td>
</tr>
</tbody>
</table>

Table 3.1: List of combinations of modalities and machine learning algorithms
3.3.4 Phase 4 – Ensemble

During this final phase, the team gathers the best combinations of modalities and machine learning algorithms from phase 3 and combines their results using ensemble learning. There are mainly 3 methods that are experimented with.

Method 1: Simple majority voting system. From the 4 models, the majority votes on the prediction will be the final model prediction.

<table>
<thead>
<tr>
<th>Videos/Classifiers</th>
<th>Frame</th>
<th>Facial</th>
<th>Audio</th>
<th>Transcription</th>
<th>Final</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predictions 1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1 (Truth)</td>
</tr>
<tr>
<td>Predictions 2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0 (Lie)</td>
</tr>
</tbody>
</table>

Table 3.2: Example of ensemble method 1

Method 2: Average voting system. The prediction probabilities of the 4 models are averaged to create the final prediction probability. If the probability is closer to 1, then it predicts “truth”. If the probability is closer to 0, then it predicts “lie”.

<table>
<thead>
<tr>
<th>Videos/Classifiers</th>
<th>Frame</th>
<th>Facial</th>
<th>Audio</th>
<th>Transcription</th>
<th>Final</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predictions 1</td>
<td>0.8</td>
<td>0.3</td>
<td>0.7</td>
<td>0.5</td>
<td>0.575 (Truth)</td>
</tr>
<tr>
<td>Predictions 2</td>
<td>0.4</td>
<td>0.2</td>
<td>0.9</td>
<td>0.1</td>
<td>0.4 (Lie)</td>
</tr>
</tbody>
</table>

Table 3.3: Example of ensemble method 2

Method 3: Weighted voting system. The prediction probabilities of the 4 models are summed together with a weight. The weight is trained from the train data and the outputs from the 4 models. If the final sum is closer to 1, then it predicts “truth”. If the final sum is closer to 0, then it predicts “lie”.

<table>
<thead>
<tr>
<th>Videos/Classifiers</th>
<th>Frame</th>
<th>Facial</th>
<th>Audio</th>
<th>Transcription</th>
<th>Final</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weights</td>
<td>0.3</td>
<td>0.3</td>
<td>0.2</td>
<td>0.2</td>
<td>/</td>
</tr>
<tr>
<td>Predictions</td>
<td>0.8</td>
<td>0.3</td>
<td>0.7</td>
<td>0.5</td>
<td>0.57 (Truth)</td>
</tr>
</tbody>
</table>

Table 3.4: Example of ensemble method 3

The effectiveness and feasibility of each method will be further discussed in the next chapter.
Chapter 4 – Experiments and Results

This chapter discusses the experiments conducted and displays the results. Then the final product’s details will be explained.

4.1 Experiments with Original Frame Module

This section discusses all experiments conducted in the original frame module. The problems encountered, reasons that may cause the problems, and the possible solutions are explored.

4.1.1 Experiment 1 – 20 Frames with VGG16

20 frames from each video were extracted using Method 1 of frame extraction (see section 3.3.2.1). The first preliminary test was conducted using the original 20 frames. The model is trained using the full VGG-16 model (see section 3.3.2.2) including its feature extraction layers and classification layers. The model is trained over 100 epochs with the last layer set to trainable.

Figure 4.1: Results of Original Frame Module Experiment 1

Figure 4.1 displays the results of experiment 1 with 20 frames per video as input. It can be observed that the training and validation accuracy both stay at around 50% throughout 100 Epochs.
- Problems: Low accuracy and lack of improvement over time.
- Hypothesis:
  - There may be misconfigured parameters in the program.
  - Algorithm may be unable to learn anything useful from the provided dataset.
  - The VGG-16 model is originally trained on the ImageNet dataset, which may mean that it cannot classify into the desired “truth” or “lie” categories.
- Possible Solution: Use the features extracted from the VGG-16 model but use another machine learning algorithm for classification.

4.1.2 Experiment 2 – 20 Frames with different ML Classifiers

20 frames from each video were extracted using Method 1 of frame extraction (see section 3.3.2.1). Considering the problems faced in experiment 1, features are now extracted using VGG-16, and the model is trained using MLP and SVM over 10 epochs. Figures 4.2 and 4.3 show the results of the initial tests.

![Training and Validation Accuracy Graph](image-url)

Figure 3.2 - Results of Original Frame Module Experiment 2 with MLP
It can be observed from Figures 4.2 and 4.3 that the training accuracies of the models are very high, with up to 95% accuracy. However, the validation accuracy is as low as 20%. This result is unsatisfactory because the low validation accuracy shows signs of model overfitting.

- Problems:
  - Decrease in validation accuracy over time.
  - Model Overfitting.

- Hypothesis: Unoptimized parameters in the classification algorithms causing the algorithms to be too complicated.

- Possible Solution: Optimize parameters by grid search.

Concluding from the experiment, the team decided to continue with improving the MLP algorithm instead of SVM because of MLP’s deep learning nature, which is more desirable for image classification.
4.1.3 Experiment 3 – Optimizing MLP Layers

Considering the problems faced in experiment 3, the team proposes to try different parameters in the MLP layers, in hopes of seeing an improvement in validation accuracy. The structure of the MLP layers are as follows:

1. Dense layer
   a. Dimensions: first
   b. Activation: ReLU

2. Dense layer
   a. Dimensions: second
   b. Activation: ReLU

3. Dense layer
   a. Dimensions: 1
   b. Activation: Sigmoid

*First* and *second* are variables that the team will experiment on using grid search. The following table shows the testing accuracies and validation accuracies for different values of *first* and *second*, after 5 and 10 epochs.

<table>
<thead>
<tr>
<th>first</th>
<th>second</th>
<th>Testing accuracy (After 5 Epochs)</th>
<th>Testing accuracy (After 10 Epochs)</th>
<th>Validation accuracy (After 5 Epochs)</th>
<th>Validation accuracy (After 10 Epochs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>100</td>
<td>0.6758</td>
<td>0.9766</td>
<td>0.3584</td>
<td>0.2656</td>
</tr>
<tr>
<td>200</td>
<td>100</td>
<td>0.7109</td>
<td>0.9336</td>
<td>0.2500</td>
<td>0.3125</td>
</tr>
<tr>
<td>500</td>
<td>100</td>
<td>0.7344</td>
<td>0.9648</td>
<td>0.4375</td>
<td>0.2500</td>
</tr>
<tr>
<td>800</td>
<td>100</td>
<td>0.7070</td>
<td>0.9180</td>
<td>0.4219</td>
<td>0.3125</td>
</tr>
<tr>
<td>1000</td>
<td>100</td>
<td>0.6289</td>
<td>0.8867</td>
<td>0.4219</td>
<td>0.2812</td>
</tr>
<tr>
<td>100</td>
<td>200</td>
<td>0.8320</td>
<td>0.8945</td>
<td>0.3906</td>
<td>0.2344</td>
</tr>
<tr>
<td>200</td>
<td>200</td>
<td>0.6016</td>
<td>0.7969</td>
<td>0.4531</td>
<td>0.2656</td>
</tr>
<tr>
<td>500</td>
<td>200</td>
<td>0.8047</td>
<td>0.9453</td>
<td>0.2656</td>
<td>0.2188</td>
</tr>
<tr>
<td>800</td>
<td>200</td>
<td>0.8164</td>
<td>0.9375</td>
<td>0.3281</td>
<td>0.2969</td>
</tr>
<tr>
<td>1000</td>
<td>200</td>
<td>0.7578</td>
<td>0.9648</td>
<td>0.4062</td>
<td>0.2656</td>
</tr>
</tbody>
</table>
It can be seen from Table H that the more epochs trained, the higher the training accuracy, which is expected. However, the validation accuracy always decreases from epochs 5 to 10, which is undesirable. The validation accuracy for all values of \( \text{first} \) and \( \text{second} \) does not once surpass 0.5. The best performing model is highlighted in Table H, with \( \text{first} = 1000 \) and \( \text{second} = 500 \), at 10 epochs trained, the testing accuracy arrives at 0.9648 and validation accuracy at 0.4062.

The team concludes that no matter the parameters in the MLP model, there is still an overfitting problem. Therefore, the problem does not lie in the parameters, but rather in the provided input data.

- Problems: Overfitting.
- Hypothesis: Not enough input data.
- Possible Solution: Data Augmentation to increase the total amount of data.
4.1.4 Experiment 4 – Data Augmentation

With 320 videos and over 1 million features, there is not enough data to properly train a deep learning neural network. Therefore, the team proposes to augment the original frames to increase the total amount of training data. The following table shows the results achieved by adding horizontally flipped original images to the training data. A similar experiment to experiment 3 is conducted.

<table>
<thead>
<tr>
<th>first</th>
<th>second</th>
<th>Testing accuracy (After 5 Epochs)</th>
<th>Testing accuracy (After 10 Epochs)</th>
<th>Validation accuracy (After 5 Epochs)</th>
<th>Validation accuracy (After 10 Epochs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>100</td>
<td>0.7129</td>
<td>0.9316</td>
<td>0.4453</td>
<td>0.4062</td>
</tr>
<tr>
<td>200</td>
<td>200</td>
<td>0.8086</td>
<td>0.9512</td>
<td>0.3986</td>
<td>0.3672</td>
</tr>
<tr>
<td>500</td>
<td>500</td>
<td>0.6309</td>
<td>0.9355</td>
<td>0.4844</td>
<td>0.3125</td>
</tr>
</tbody>
</table>

Table 4.2: Accuracy results of Original Frame Module Experiment 4

Table 4.1 (Extract): Accuracy results of Original Frame Module Experiment 3

Table 4.2 shows the accuracy results of experiment 4 with 3 combinations of first and second variables. The extract of experiment 3’s results are also shown for comparison. Experiment 4’s validation accuracies is higher than that of experiment 3 in general. This shows that by increasing the amount of training data, the overfitting problem becomes less severe. However, the validation accuracies after 10 epochs still cannot reach 0.5, which is still unsatisfactory.

The team proposes to do more image augmentation but is quickly met with the limitation that doing so would take too much time and storage space. So, new methods to increase training data must be explored.

- Problems:
  - Overfitting.
  - Image augmentation is too expensive.
• Hypothesis: Training data to features ratio is imbalanced.
• Possible Solution: Explore new methods to increase training data.

4.1.5 Experiment 5 – New Frame Extraction Method & PCA

Experiment 3-5 has concluded that the lack of training data is a major cause of the model’s overfitting problem. To increase the amount of training data, a new method for extracting frames is developed, specifically, method 2 of frame extraction (see section 3.3.2.1). Each video is translated into 20 sets of data, each containing 10 sample frames of the video. The project team successfully extracted their feature vectors using the VGG-16 model.

Principal components analysis (PCA) is also conducted on the feature vectors to reduce the input dimension. The following table shows the cumulated explained variance ratio for the tested dimensions.

<table>
<thead>
<tr>
<th>n_components</th>
<th>Cumulative explained variance ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>25088 (Original dimensions)</td>
<td>1.0</td>
</tr>
<tr>
<td>3200</td>
<td>0.905</td>
</tr>
<tr>
<td>4096</td>
<td>0.931</td>
</tr>
<tr>
<td>5000</td>
<td>0.950</td>
</tr>
</tbody>
</table>

Table 4.3: The accumulated explained variance for each PCA parameter value (Original Frame)

The team proposes to use as little components as possible while keeping an acceptable value of explained variance ratio. For this project, at least 0.9 variance is kept. Therefore, the optimal PCA parameter is chosen as n_components = 3200. In other words, the original 25088 dimensions per frame is reduced to 3200 dimensions per frame, while keeping 90.5% variance.

After PCA, the features from the 10 frames in each set are concatenated together to form a final input dimension of 32000.

As a result, 6400 training data is extracted from 320 original videos while the input dimension is reduced to 32000. Compared to the previous method, the number of training data has increased by 20 times, and the dimensions per input has reduced by 7.84 times. This should reduce the severity of the imbalance between the training data and features problem.
4.1.6 Experiment 6 – Final Frame Model
With the newly reshaped training data, the team retrained the model with SVM, KNN, Random Forest and MLP.

SVM
Using the default parameters for SVM in sklearn’s SVM module, the model achieved an average accuracy of 0.49 over 10-fold cross validation. Compared to experiment 2 (see section 4.1.2), the accuracy has improved, but is still worse than random guessing. Therefore, the result is unsatisfactory.

Random Forest
Different parameters were tested in the Random Forest Classifier, namely the max depth and number of trees. Maximum features is set to 3. The following figure shows the results achieved.

![Figure 4.4: Results of Random Forest Classifier in Original Frame Module Experiment 6](image)

The best accuracy achieved by Random Forest Classifier is 57.5% with 20 trees and max depth of 5. This is an 8.5% improvement compared to the SVM Classifier. This classifier is considered a success and may be tested again in future experiments.
KNN

Different parameter values were tested in the KNN Classifier, namely the number of neighbors. The following figure shows the results achieved.

Figure 4.5: Results of KNN Classifier in Original Frame Module Experiment 6

The best accuracy achieved by KNN Classifier is 53% with 15 and 20 neighbors. This is a 4% improvement compared to the SVM Classifier. However, the KNN Classifier does not perform as well as the Random Forest Classifier. Therefore, this classification method will not be further explored.
MLP

Different parameters were tested in the MLP Classifier, namely the number of perceptrons per layer, and the number of epochs. Through brute force searching, the team found the best structure is as follows:

1. Dense:
   a. Dimensions: 100
   b. Activation: ReLU
2. Dense:
   a. Dimensions: 30
   b. Activation: ReLU
3. Dense:
   a. Dimensions: 1
   b. Activation: Sigmoid

The result is shown in figure 4.6.

Figure 4.6: Results of MLP Classifier in Original Frame Module Experiment 6

The validation accuracy has increased to about 58%, which is better than the previous MLP experiments which had never surpassed 50%. Although there are still signs of overfitting, the team decided that the accuracy is acceptable, compared to the other existing studies achieving 56.8% accuracy with video data only [2]. Considering this is the best accuracy achieved from all Original Frame experiments, the team will be including this MLP model in the final product.
4.2 Experiments with Facial Image Module
This section discusses all experiments conducted in the facial image module. The problems encountered, reasons that may cause the problems, and the possible solutions are explored.

4.2.1 Experiment 1 – Expression Recognition
The team first hypothesizes that the expressions of an individual can indicate whether that person is lying or not. Therefore, method 2 of facial pre-processing (see section 3.3.2.4) is used in this experiment to extract the facial expressions for each frame in each video. The expressions are classified with a pre-trained expression recognition model built by WuJie1010 [T]. The facial images are classified into 7 expressions, namely, anger, disgust, fear, happy, sad, surprise, and neutral. The score of the output is used to train the lie detection classifier. Different classifiers were tested with 5-fold cross validation. The results are as follows.

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Average Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logistic Regression</td>
<td>53.1%</td>
</tr>
<tr>
<td>SVM</td>
<td>50.9%</td>
</tr>
<tr>
<td>Decision Tree</td>
<td>51.6%</td>
</tr>
<tr>
<td>AdaBoost</td>
<td>47.6%</td>
</tr>
<tr>
<td>Random Forest</td>
<td>54.6%</td>
</tr>
<tr>
<td>KNN</td>
<td>53.4%</td>
</tr>
<tr>
<td>MLP</td>
<td>54.9%</td>
</tr>
</tbody>
</table>

Table 4.4: Accuracy results of Facial Image Experiment 1

Table 4.4 lists the average accuracy of different classifiers. The MLP Classifier performed best at 54.9% accuracy. However, this accuracy is only slightly better than random guessing. Therefore, the team concludes that this method of facial pre-processing is not suitable.

- Problem: Low accuracy.
- Hypothesis: The score from expressions recognition is not learnable.
- Possible Solution: Try another facial pre-processing method.
4.2.2 Experiment 2 – New Facial Extraction Method & PCA

Experiment 2 uses a similar method to original frame module’s experiment 5 (see section 4.2.5). The facial images are first extracted using method 1 of facial extraction (see section 3.3.2.4). Each video is translated into 20 sets, with 10 facial images in each set. The project team successfully extracted their feature vectors using the VGG-16 model.

Principal components analysis (PCA) is also conducted on the feature vectors to reduce the input dimension. The following table shows the cumulated explained variance ratio for the tested dimensions.

<table>
<thead>
<tr>
<th>n_components</th>
<th>Cumulative explained variance ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>25088 (Original dimensions)</td>
<td>1.0</td>
</tr>
<tr>
<td>3200</td>
<td>0.816</td>
</tr>
<tr>
<td>4096</td>
<td>0.864</td>
</tr>
<tr>
<td>6000</td>
<td>0.904</td>
</tr>
</tbody>
</table>

Table 4.5: The accumulated explained variance for each PCA value (Facial Image)

Like the original frame module experiment 5, the team proposes to keep at least 0.9 variance. Therefore, the optimal PCA parameter is chosen as n_components = 6000. In other words, the original 25088 dimensions per frame is reduced to 6000 dimensions per frame, while keeping 90.4% variance. After PCA, the features from the 10 frames in each set are concatenated together to form a final input dimension of 60000.

As a result, 6080 training data is extracted from 304 original videos* while the input dimension is reduced to 60000. Compared to the previous method, the number of training data has increased by 20 times, and the dimensions per input has reduced by 4.18 times. This should reduce the severity of the imbalance between the training data and features problem.

* Note that if any subjects’ video has less the 200 faces extracted, all videos from that subject are discarded. In this case, 4 subjects / 16 videos were discarded. It is implemented this way to keep the training data having the same number of “truths” and “lies” data.
4.2.3 Experiment 3 – Final Facial Model

In this experiment, the team trains the facial data with Random Forest, KNN, and MLP.

KNN

Different parameter values were tested in the KNN Classifier, namely the number of neighbors. The following figure shows the results achieved.

![Accuracy and Number of neighbors](image)

Figure 4.7: Results of KNN Classifier in Facial Image Module Experiment 3

The best accuracy achieved by KNN Classifier is 50.4% with 31 neighbors. This result is unsatisfactory as the accuracy is only 0.4% better than random guessing. Therefore, this classification method will not be further explored.
Random Forest

Different parameters were tested in the Random Forest Classifier, namely the max depth and number of trees. Maximum features is set to 3. The following figure shows the results achieved.

![Accuracy and Number of Trees in the forest](image)

Figure 4.8: Results of Random Forest Classifier in Facial Image Module Experiment 3

The best accuracy achieved by Random Forest Classifier is 54.25% with 10 trees and max depth of 1. This is a 3.75% improvement compared to the KNN Classifier. However, this accuracy is only slightly better than random guessing, therefore this classifier will not be included in the final product.
MLP

Different parameters were tested in the MLP Classifier, namely the number of perceptrons per layer, and the number of epochs. Through brute force searching, the team found the best structure is as follows:

1. Dense:
   a. Dimensions: 50
   b. Activation: ReLU

2. Dense:
   a. Dimensions: 10
   b. Activation: ReLU

3. Dense:
   a. Dimensions: 1
   b. Activation: Sigmoid

The result is shown in figure 4.9.

![Training and Validation Accuracy](image)

**Figure 4.9: Results of MLP Classifier in Facial Image Module Experiment 3**

The validation accuracy is 54.92%, which is slightly better than the KNN and Random Forest Classifiers. Considering this is the best accuracy achieved so far from all Facial Image experiments, the team will be including this MLP model in the final product.
4.3 Experiments with Audio Module

This section discusses all experiments conducted in the audio module. The problems encountered, reasons that may cause the problems, and the possible solutions are explored.

4.3.1 Experiment 1 - KNN

With the MFCCs feature extracted from the audio clips (see section 3.3.2.5), a KNN Classifier is trained to classify between “truth” and “lie”. Different numbers of neighbors were tested in the classifier. The following figure shows the results achieved.

![Figure 4.10: Results of KNN Classifier in Audio Module Experiment 1](image)

The best accuracy achieved by KNN Classifier is 62.5% with 1 neighbor. Although the accuracy is the highest the project has achieved so far, a 1 neighbor KNN Classifier is quite unconventional because of its low bias. It is essentially overfitting to the training data. The second-best accuracy is 50% with 11 neighbors, which is unsatisfactory. Therefore, this KNN model will not be used in the final model.

- Problem: Although decent accuracy, there is overfitting.
- Possible Solution: Try another classifier.
4.3.2 Experiment 2 – Random Forest

With the MFCCs feature extracted from the audio clips (see section 3.3.2.5), a Random Forest Classifier is trained to classify between “truth” and “lie”. Different parameters were tested in the Random Forest Classifier, namely the max depth and number of trees. Maximum features is set to 3. The following figure shows the results achieved.

![Accuracy and Number of Trees in the forest](image)

**Figure 4.11:** Results of Random Forest Classifier in Audio Module Experiment 2

The best accuracy achieved by Random Forest Classifier is 67.2% with 30 trees and max depth of 5. This is by far the best accuracy the team has achieved for lie detection and the team is happy with this performance. Therefore, this classifier will be included in the final product.
4.4 Experiments with Text Module

This section discusses all experiments conducted in the text module. The reasoning behind the experiments and the conclusions are explored.

4.4.1 Experiment 1 – Count Vectorizing

This experiment converts the text data into a matrix of token counts using sklearn’s CountVectorizer class. A brief overview of the project’s text pre-processing can be found in section 3.3.2.6. The “ngram_range” parameter in the CountVectorizer class depicts the lower and upper boundary of the range of n-values for different word n-grams to be extracted. For example, ngram_range of (1,1) means only unigrams, (1,2) means unigrams and bigrams combined, and (2,2) means only bigrams [X]. For the project’s purposes, the values [(1,1),(1,2),(2,2)] for ngram_range will be tested.

The following figure shows the top 20 most frequent words in ngram_range (1,1).

![Top 20 Most Frequent Words](image)

Common words like “and”, “to” and “like” appear the most frequently. Since the MU3D dataset’s transcription also includes the subjects’ stuttering words, the word “um” is at the 4th most frequently said words. The team hypothesis that these stutter words can be very useful in determining whether the subject is lying or not.

The following figure shows the top 20 most frequent pair of words in ngram_range (2,2).
The most frequent pairs of words found are “you know” and “kind of”, which are casual words pairs when speaking naturally. It can be observed that the common stutter word “um” appears several times in words pairs such as “um she”, “um he”, “and um” and “um and”. The team believes that these pairs would also be useful in classifying lies.

To find out which parameter value produces the best accuracy, the team experimented on the settings with 10-fold cross validation. The following table shows the results.

<table>
<thead>
<tr>
<th>Ngram_range</th>
<th>Number of times as best setting</th>
<th>Average Accuracy as best setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1,1)</td>
<td>4</td>
<td>62.5%</td>
</tr>
<tr>
<td>(1,2)</td>
<td>6</td>
<td>62.3%</td>
</tr>
<tr>
<td>(2,2)</td>
<td>0</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 4.6: The results for different ngram_range values in Text Module Experiment 1

Table 4.6 shows the number of times a ngram_range value was picked as the best setting during the 10-fold cross validation. Although (1,1) has a higher accuracy with a 0.2% advantage over (1,2), (1,2) appeared most as the best setting. The (2,2) setting did not appear once as best setting, clearly showing that it is not optimal. Therefore, the team chooses the ngram_range setting as (1,2) in future experiments.
4.4.2 Experiment 2 – Normalizing Term Frequency

The TfidfTransformer from sklearn is used for normalizing the term frequencies. The goal of using tf-idf is to scale down the impact of tokens that occur very frequently in a given corpus and are less informative than features that occur in a small fraction of the training corpus [X]. The following table compares the accuracy when classifying with TfidfTransformer to without in a 10-fold cross validation.

<table>
<thead>
<tr>
<th>Use_idf</th>
<th>Number of times as best setting</th>
<th>Average Accuracy as best setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>6</td>
<td>62.0%</td>
</tr>
<tr>
<td>False</td>
<td>4</td>
<td>63.0%</td>
</tr>
</tbody>
</table>

Table 4.7: The results for different use_idf values in Text Module Experiment 2

Table 4.7 shows the number of times the tfidf algorithm was used as the best setting during the 10-fold cross validation. Although not using the algorithm yielded a better accuracy as best setting with a 1% advantage, using the tfidf algorithm is better because it appeared as best setting more often. Therefore, the team will continue to use the tfidf algorithm in future experiments.

4.4.3 Experiment 3 – SVM

From previous experiences and knowledge, the team chose SVM for classifying the transcript. The following figure shows the pipeline for classifying text.

Figure 4.14: Pipeline for Text Module Experiment 3

Combining with the experiment results in experiment 1 and 2, the SVM model reached an accuracy of **62.5%**. The team is satisfied with this result, and therefore will be using this model in the final product.
4.5 Summary of Module Results

This section summarizes the above modules’ results. The following table shows the best accuracy achieved for each combination of modality and classifiers.

<table>
<thead>
<tr>
<th>Modality</th>
<th>Machine Learning Algorithm</th>
<th>Best Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original Image</td>
<td>Multilayer Perceptron (MLP)</td>
<td>58%</td>
</tr>
<tr>
<td></td>
<td>VGG-16</td>
<td>52%</td>
</tr>
<tr>
<td></td>
<td>Support Vector Machine (SVM)</td>
<td>49%</td>
</tr>
<tr>
<td></td>
<td>Random Forest</td>
<td>57.5%</td>
</tr>
<tr>
<td></td>
<td>KNN</td>
<td>53%</td>
</tr>
<tr>
<td>Facial Image</td>
<td>Multilayer Perceptron (MLP)</td>
<td>54.9%</td>
</tr>
<tr>
<td></td>
<td>Logistic Regression</td>
<td>53.1%</td>
</tr>
<tr>
<td></td>
<td>Support Vector Machine (SVM)</td>
<td>50.9%</td>
</tr>
<tr>
<td></td>
<td>Decision Tree</td>
<td>51.6%</td>
</tr>
<tr>
<td></td>
<td>KNN</td>
<td>53.4%</td>
</tr>
<tr>
<td></td>
<td>AdaBoost</td>
<td>47.6%</td>
</tr>
<tr>
<td></td>
<td>Random Forest</td>
<td>54.6%</td>
</tr>
<tr>
<td>Audio</td>
<td>Random Forest</td>
<td>67.2%</td>
</tr>
<tr>
<td></td>
<td>KNN</td>
<td>62.5%</td>
</tr>
<tr>
<td>Transcription</td>
<td>Support Vector Machine (SVM)</td>
<td>62.5%</td>
</tr>
</tbody>
</table>

Table 4.8: Results of combinations of modalities and machine learning algorithms

Table 4.8 shows all the results of the combinations of modalities and machine learning algorithm achieved in the previous experiments. The highlighted combinations are chosen to be included in the final product. These combinations will be further combined via ensemble learning, which will be discussed in the next section. The results also suggest that audio and text are better modalities for determining lies than images.
4.6 Experiments with Ensemble

This section discusses all experiments conducted during the ensemble phase. The problems encountered, reasons that may cause the problems, and the possible solutions are explored.

4.6.1 Experiment 1 – Simple Majority Voting

To combine the results from the 4 final classifiers from the 4 modules, namely Original Frame Module, Facial Image Module, Audio Module and Text Module, a voting system must be in place. Method 1 of majority voting is proposed (see section 3.3.4). However, this method is not feasible because there are even numbers of models. If 2 models predict “lie” and 2 models predict “truth”, then the final voting system cannot choose between “truth” or “lie”. The following table depicts the problem.

<table>
<thead>
<tr>
<th>Videos\Classifiers</th>
<th>Frame</th>
<th>Facial</th>
<th>Audio</th>
<th>Transcription</th>
<th>Final</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predictions 1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1 (Truth)</td>
</tr>
<tr>
<td>Predictions 2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0 (Lie)</td>
</tr>
<tr>
<td>Predictions 3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>?</td>
</tr>
</tbody>
</table>

Table 4.9: Failed example of ensemble method 1

In the case of prediction 3, the final model cannot generate an accurate prediction. If the prediction is hardcoded to predict only “truth” or only “lie”, then the final accuracy would be low.

- Problem: Issue when there are equal numbers of “truths” and “lies” in module models.
- Possible Solution: Use more precise predictions in the module models.
4.6.2 Experiment 2 – Average Majority Voting

This experiment uses Method 2 of ensemble (see section 3.3.4). The prediction probabilities of the 4 module’s models are averaged to create the final prediction probability. The following figure shows a snippet of the average voting in action.

![Figure 4.15: The results of Ensemble Experiment 2](image)

The final accuracy of Method 2 arrives at **60.94%**. This accuracy is an improvement compared to image modules but is comparatively worse than audio and text modules.

- **Problems**: The final accuracy is lower than expected.
- **Hypothesis**: Some module models are more informative than others, using average to calculate final prediction may not be optimal.
- **Possible Solution**: Weighted average approach,
4.6.3 Experiment 3 – Weighted Average Voting (Perceptron)

This experiment uses Method 3 of ensemble (see section 3.3.4). The weight for each module model is trained from the training data. The team proposes to use a simple perceptron algorithm to train the weights. The following table shows the coefficient for each model.

<table>
<thead>
<tr>
<th></th>
<th>Frame</th>
<th>Facial</th>
<th>Audio</th>
<th>Transcription</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coefficient / Weight</td>
<td>0.433</td>
<td>0.447</td>
<td>0.042</td>
<td>0.079</td>
</tr>
</tbody>
</table>

Table 4.10: The trained weights for each module model

The final accuracy of Method 3 arrives at 64.1%, which is better than Method 2. However, it is illogical that the weights for frame and facial take up majority of the proportion, since the accuracy of frame and facial models are lower than that of audio and text models.

- Problem: The weights are skewed towards frame and facial models even though they have lower model accuracies.
- Hypothesis: The prediction for training data in frame and facial models have very high prediction probability, up to 99%, because the training accuracy is very high for both models. Therefore, the perceptron in method 3 will train the weights to be very skewed towards frame and facial models.
- Possible Solution: Use unseen data to train the weights instead of the original training data.

However, the proposed solution is unfeasible in our case. This is because there is not enough data to split the testing data even more for training. Therefore, because of the severe overfitting nature of this method, the team decided to not use this method despite its higher accuracy compared to method 2.
4.7 Module Ensemble Analysis

This section analyzes the effect of different combinations of modalities for ensemble. The following tables show the relationship between different modules and how they perform when combined.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Original Frame</th>
<th>Facial Image</th>
<th>Audio</th>
<th>Transcript</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original Frame</td>
<td>1.0</td>
<td>0.5</td>
<td>0.6389</td>
<td>0.4722</td>
<td></td>
</tr>
<tr>
<td>Facial Image</td>
<td>0.5</td>
<td>1.0</td>
<td>0.6389</td>
<td>0.6667</td>
<td></td>
</tr>
<tr>
<td>Audio</td>
<td>0.5349</td>
<td>0.5349</td>
<td>1.0</td>
<td>0.4651</td>
<td></td>
</tr>
<tr>
<td>Transcript</td>
<td>0.5667</td>
<td>0.8</td>
<td>0.6667</td>
<td>1.0</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.11: Relationship between Classifiers 1

Table 4.11 shows the relationship between Classifiers A & B, where the value represents the percentage of test data Classifier B was able to classify correctly given Classifier A classified correctly. The value means how many correct predictions of Classifier A overlaps with Classifier B. For example, Transcript VS Facial Image has a value of 0.8, which means that 80% of correct predictions by Transcript Classifier is also predicted correctly by Facial Image Classifier. This shows that combining Facial Image Classifier with Transcript Classifier will not improve the final classifier by a large amount.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Original Frame</th>
<th>Facial Image</th>
<th>Audio</th>
<th>Transcript</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original Frame</td>
<td>0.0</td>
<td>0.6429</td>
<td>0.7143</td>
<td>0.4643</td>
<td></td>
</tr>
<tr>
<td>Facial Image</td>
<td>0.6429</td>
<td>0.0</td>
<td>0.7143</td>
<td>0.2143</td>
<td></td>
</tr>
<tr>
<td>Audio</td>
<td>0.6190</td>
<td>0.6190</td>
<td>0.0</td>
<td>0.4762</td>
<td></td>
</tr>
<tr>
<td>Transcript</td>
<td>0.5588</td>
<td>0.3529</td>
<td>0.6765</td>
<td>0.0</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.12: Relationship between Classifiers 2

Table 4.12 shows the relationship between Classifiers A & B, where the value represents the percentage of test data Classifier B was able to classify correctly given Classifier A classified wrongly. The higher the value means Classifier B can improve the final prediction by a larger amount when it is combined with Classify A. This shows that when Original Frame Classifier or Facial Image Classifier is combined with Audio classifier, it will improve the final accuracy.
<table>
<thead>
<tr>
<th>Modality</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Only Original Image</td>
<td>58%</td>
</tr>
<tr>
<td>Only Facial Image</td>
<td>54.9%</td>
</tr>
<tr>
<td>Only Audio</td>
<td>67.2%</td>
</tr>
<tr>
<td>Only Transcript</td>
<td>62.5%</td>
</tr>
<tr>
<td>Original Image + Facial Image</td>
<td>56.25%</td>
</tr>
<tr>
<td>Original Image + Audio</td>
<td>59.375%</td>
</tr>
<tr>
<td>Original Image + Transcript</td>
<td>54.6875%</td>
</tr>
<tr>
<td>Facial Image + Audio</td>
<td>56.25%</td>
</tr>
<tr>
<td>Facial Image + Transcript</td>
<td>56.25%</td>
</tr>
<tr>
<td>Audio + Transcript</td>
<td>65.625%</td>
</tr>
<tr>
<td>Original Image + Facial Image + Audio</td>
<td>59.375%</td>
</tr>
<tr>
<td>Original Image + Facial Image + Transcript</td>
<td>57.8125%</td>
</tr>
<tr>
<td>Original Image + Audio + Transcript</td>
<td>59.375%</td>
</tr>
<tr>
<td>Facial Image + Audio + Transcript</td>
<td>56.25%</td>
</tr>
<tr>
<td>All Four</td>
<td>60.9375%</td>
</tr>
</tbody>
</table>

Table 4.13: Accuracy of different combinations of modalities

Table 4.13 shows the accuracy of each ensemble model when combining with different modalities. There are 2 combinations that broke past 60% accuracy, namely Audio + Transcript and All Four. To closely follow the project’s objectives, the team will use all four modalities for the final product.
4.8 Final Product

This section covers the final product containing the final model and its interactive GUI.

4.8.1 GUI

The final product should be presentable and interactive. Users should be able to input any video into the program, and the program will give a prediction between “Truth” or “Lie”. The GUI has the following inputs.

1. Data type: To choose between using the provided MU3D videos or new videos.
2. File name: A list of all MU3D videos (applicable if Existing data is selected in Data type)
3. Folder address: The location of the folder containing the new video and transcription. (only applicable for new data)
4. New video file name: The new video file name (only applicable for new data, must include file extension e.g. .wmv)
5. New text file name: The transcription file name (only applicable for new data)
6. Start: To send the input to the classifier and start classifying.
7. Reset output: To clear the input and output fields.

An example of the output is displayed in section 4.6.3.
4.8.2 Integration

With all models and ensemble methods tested, everything must be combined into a single program that takes a video input and outputs the prediction. Therefore, the program includes these functions.

1. Original Frame Module
   a. Frame Extraction
   b. Frame Pre-processing
   c. Frame Classification

2. Facial Image Module
   a. Facial Image Extraction
   b. Facial Image Pre-processing
   c. Facial Classification

3. Audio Module
   a. Audio Extraction
   b. Audio Pre-processing
   c. Audio Classification

4. Text Module
   a. Text Extraction
   b. Text Pre-processing
   c. Text Classification

5. Ensemble

Since all the models are already trained, the extraction process for frames and facial images are different. In frame extraction (function 1a), like Method 1 of frame extraction (see section 3.3.2.1), the video is divided into 20 buckets equally instead of 10. Then the first frame from each bucket will be extracted to form the set. In facial image extraction, like frame extraction, the video is divided into 20 buckets equally. Since faces may not be detected for every frame, the first face that can be extracted from each bucket will be extracted to form the set. If one of the buckets does not contain a face can be extracted, then the whole facial image module will not be included in the final classification.
Important note for function 4a, since the bare minimum input is a video, sometimes the transcript of the video is not supplied. To solve this problem, the team proposes to use a speech recognition model developed by AssemblyAI [21]. Then after generating the transcription, the text can be piped into function 4b and onwards.

Since the program runs on Google Colaboratory, connection to the internet must be stable. Internet connection is also needed to connect to AssemblyAI’s API.

An example of an output is displayed in the next section.

4.8.3 Product Example
This section shows an example input and output of the final product.

Input: BF025_2NL.wmv

Output:

---Audio Preprocessing In Progress...---
BF025_2NL.wmv /content/drive/MyDrive/MU3D/Videos/BF025_2NL.wmv
MoviePy - Writing audio in /content/BF025_2NL.wav
chunk:  0% | 0/791 [00:00<?, ?it/s, now=None]
chunk: 39% | 306/791 [00:00<00:00, 3016.64it/s, now=None]
chunk: 77% | 612/791 [00:00<00:00, 2984.77it/s, now=None]
MoviePy - Done.
---Audio Preprocessing Completed.---
---Frame/Face Extraction In Progress...---
1075 total frames in BF025_2NL.wmv

Frames Extracted from BF025_2NL.wmv
The model has predicted 0, which means it is a “Lie”. Since the input video “WF024_2NL” is of type “Negative Lie”, the model has predicted correctly!

4.8.4 Product Conclusion

In conclusion, this product takes video and its transcription as input. It separates the inputs into 4 categories, namely original frame, facial image, audio, and transcription. Then, it uses 4 different classifier modules to classify the 4 datasets (Original Frame Classifier, Facial Image Classifier, Audio Classifier, and Transcription Classifier). Lastly, the results of the classifiers are combined by ensemble learning. The final accuracy of 60.2% is achieved.
Chapter 5 – Conclusion

This chapter concludes the Lie Detector with Machine Learning project, reinstating the objectives and results of the project. Finally, the project’s limitations are discussed, including some proposed solutions with future works.

5.1 Context and Aims
Lie detection with machine learning has become a popular topic nowadays. This project proposes to create a machine learning model for lie detection while extending previous work on this topic. The team will apply several data types and classification techniques to find the best combinations and combine them into the final model using ensemble learning.

5.2 Comments on Results
Overall, the results are satisfactory. Although the model did not achieve ground-breaking accurate with 60.9%, the team is happy with how the final product turned out. It is essentially a program that takes the video input and automatically does all feature extraction and pre-processing in real-time and classifies the video into “truth” or “lie”. It is quite interactive and would be fun to play around with.

5.3 Limitations and Future Work
The team chose videos as the base dataset because it is more easily accessible than advanced data types. Therefore, advanced data types like eye gaze technology are out of this project’s scope. However, because the project implements ensemble learning, it is not difficult to add additional modalities to improve the model in the future. It is recommended that future researchers should build upon this project and continue to add more functionalities and modalities if they choose to. In addition to adding more modalities, more training data would help relieve the overfitting problem that is faced throughout the whole project.

5.4 Contributions
The project’s model can take a video of a person speaking as input and give a prediction of whether that person is lying or not. As of now, no product with this functionality exists. Additionally, the project’s paper will provide an in-depth analysis of which combination of data types and algorithms works best with lie detection. Therefore, this project will be a significant steppingstone in discovering more about lie detection with machine learning.
References

[1] C. Melore, “You look great!' average person tells 4 lies per day, survey shows,”

[2] B. Steinhilber, "How to tell if someone is lying to you, according to researchers,"


## Appendix A – Python Libraries

<table>
<thead>
<tr>
<th>Library</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scikit</td>
<td>Pre-built machine learning models and preprocessing functions</td>
</tr>
<tr>
<td>TensorFlow</td>
<td>Pre-built machine learning models</td>
</tr>
<tr>
<td>Keras</td>
<td>Pre-built machine learning models</td>
</tr>
<tr>
<td>NumPy</td>
<td>Calculations</td>
</tr>
<tr>
<td>pandas</td>
<td>Data manipulation</td>
</tr>
<tr>
<td>cv2</td>
<td>Video reading, frame extraction</td>
</tr>
<tr>
<td>re</td>
<td>Regular Expressions</td>
</tr>
<tr>
<td>os</td>
<td>Saving and loading files</td>
</tr>
<tr>
<td>import_ipynb</td>
<td>Importing other python files from google drive</td>
</tr>
<tr>
<td>autocrop</td>
<td>Facial image extraction</td>
</tr>
<tr>
<td>matplotlib</td>
<td>Visualization</td>
</tr>
<tr>
<td>librosa</td>
<td>Audio Feature extraction</td>
</tr>
<tr>
<td>resampy</td>
<td>Audio Feature extraction</td>
</tr>
<tr>
<td>moviepy</td>
<td>Audio extraction</td>
</tr>
<tr>
<td>joblib</td>
<td>Saving and loading models</td>
</tr>
<tr>
<td>requests</td>
<td>Html requests for speech recognition</td>
</tr>
</tbody>
</table>

Table A: List of python libraries used during development.